Rapid Decimation for Direct Volume Rendering

Jonathan Gibbs, Allen Van Gelder, Vivek Verma, and Jane Wilhelms
University of California, Santa Cruz
UCSC–CRL–97-26

December 19, 1997

Abstract

An approach for eliminating unnecessary portions of a volume when producing a direct volume rendering is described. This reduction in volume size sacrifices some image quality in the interest of rendering speed. Since volume visualization is often used as an exploratory visualization technique, it is important to reduce rendering times, so the user can effectively explore the volume. The methods presented can speed up rendering by factors of 2 to 3 with minor image degradation.

A family of decimation algorithms to reduce the number of primitives in the volume without altering the volume's grid in any way is introduced. This allows the decimation to be computed rapidly, making it easier to change decimation levels on the fly. Further, because very little extra space is required, this method is suitable for the very large volumes that are becoming common. The method is also grid-independent, so it is suitable for multiple overlapping curvilinear and unstructured, as well as regular, grids. The decimation process can proceed automatically, or can be guided by the user so that important regions of the volume are decimated less than unimportant regions.

A formal error measure is described based on a three-dimensional analog of the Radon transform. Decimation methods are evaluated based on this metric and on direct comparison with reference images.

Keywords: Computer Graphics, Scientific Visualization, Direct Volume Rendering, Decimation, Level-of-detail, Irregular Grids.

1 Introduction

Direct volume rendering is an attractive visualization technique because it can convey a lot of information in a single image. This is done by mapping the data values directly to color and opacity. Direct volume rendering is ideal for initial explorations of new data sets because it requires minimal knowledge of the data values to produce meaningful images. However, rendering an image is computationally expensive, particularly when the data is not on a regular grid. Large data sets are becoming very common due to the rapid increase in CPU speed, and the increasing affordability of 3D data acquisition techniques. The data calculation can be done off-line, but the visualization demands interactivity. Although many advances have been made in direct volume rendering in the last several years, interactivity is possible only with small data sets, and often only if they are on regular grids. Work in computation fluid dynamics, such as NASA's study of the space shuttle, is done on multiple overlapping curvilinear grids or large unstructured grids. Even with state-of-the-art hardware and algorithms, direct volume rendering of these data sets is far from interactive.

A solution to this problem is to use partial data sets for interactive exploration, and then switch to the complete data set when more details are necessary. This paper presents a process whereby the original, large

data set can be rapidly decimated for faster rendering, producing images with few visual artifacts. Since large data sets can be hundreds of megabytes, it is vital that this method introduce a minimal amount of new data. This means that the data cannot be re-sampled, or re-gridded. Since the data set is not altered in memory, there is no penalty for switching back and forth from decimated to non-decimated volumes. This paper is organized as follows:

- Background in volume rendering and decimation is discussed in section 2.
- We present a method of rendering decimated grids without re-meshing or re-gridding. This requires a direct volume renderer that can handle arbitrary grids. To achieve this, we deal exclusively with polygons instead of cells, using a volume rendering approach described previously [WVGTG96]. This polygon-based rendering technique and the general decimation algorithm are described in sections 3 and 3.1.
- In section 3.2, we present several methods of guiding the decimation. The decimation can be generated automatically, but it is often advantageous to allow the user to indicate important data values to help guide the decimation.
- We also present an image-based method for quantifying the error introduced in the volume at any given level of detail. This is covered in section 4.
- Experimental results are discussed in section 5.

2 Background and Related Work

We will first briefly discuss direct volume rendering. Next, we will survey the literature on decimation, covering surface decimation techniques first, and then discussing the volume-related work.

2.1 Volume Rendering

Early approaches for direct volume rendering used *ray-casting*, *cell projection*, and *splatting* [FDFH90]. Most research has addressed only rectilinear (regular) grids, and most previously reported acceleration and optimization techniques apply only to such grids. New methods such as Fourier transforms [Lev92, Mal93], shear-warp transforms [LL94], and 3D texture maps [CCF94] suffer this limitation.

However, many applications create non-rectilinear volume data sets, such as computational fluid dynamics (CFD), finite element analysis (FEM), and atmospheric and oceanographic measurements. Such data is often found on *curvilinear grids* (where a computational regular grid is warped to fit around objects of interest), and *unstructured grids* (where data points are connected to form tetrahedral or other polyhedral cells). Sometimes non-tetrahedral cells are broken into tetrahedra to simplify processing; however, this can lead to artifacts and increases the number of primitives that the renderer has to handle. Multiple, overlapping, and intersecting grids may be used to sample space around very complex shapes [BCFM+89]. Our research concentrates on rendering such irregular data.

A number of algorithms have been developed for irregular grids. Ray-casting general irregular grids is complicated and slow, though it does parallelize beautifully [Gar90, Cha92, Use93, Ma95]. Cell projection and splatting have been used for irregular grids in software [MHC90, Gie92, Koy92, Wil92, GP93, MHK95, SMK96] and hardware [ST90, VGW93, YRL+96]. Instead of projecting cells, several algorithms have been developed which project faces [Luc92, Cha93, WVGTG96].

2.2 Surface Decimation

While there has not been a lot of work in the decimation of volumes, there have been many papers addressing the issue of decimation of surface meshes. There are several applications which typically create very large surface meshes, such as those that generate isosurfaces from large volumes or acquire surfaces from real-world objects via 3D scanning.

Surface simplification techniques take a surface mesh comprised of a large number of polygons, and attempt to build a new model with fewer polygons, whose surface deviates as little as possible from the original. Some methods require the surface to be re-sampled, so that the new model does not include vertices from the old. Most methods also require re-meshing, so the new connections are not a subset of the old.

The simplest type of surface to decimate is the height field. This is essentially only a 2D problem. Garland and Heckbert present an overview of these techniques [GH95], most of which center around creating a Delauney triangularization of the data at various error levels. More recently, work in this area has involved generating the display in real-time and maintaining continuity between different levels of detail [LKR+96].

When dealing with an arbitrary surface, the problem becomes more complicated because the surface cannot be easily parameterized in a 2D world space. DeHaumer and Zyda proposed two solutions: either start with a coarse grid, and incrementally refine it until a criterion is met (*adaptive subdivision*), or start with the finest data, and group small polygons into larger ones (*polygon growth*) [DZ91]. Later algorithms have built on these two fundamental concepts.

Turk presented a method called *re-tiling* [Tur92]. Re-tiling re-samples the surface uniformly, and adds new vertices to the original mesh. The old vertices are then removed one by one, making local adjustments to preserve the topology of the surface. Other algorithms have been presented which are also based on the incremental removal of vertices, followed by a re-triangulation [SZL92, KT96]. Some recent vertex-removal algorithms have focused on more sophisticated error control [CCMS96, KLS96]. Other methods group planar polygons, and replace them with fewer, larger, polygons [HH93, RB93], or remove polygons by iteratively collapsing edges [HDD+93, Hop96]. Recent edge-collapsing algorithms have addressed edge selection, and allow the mesh to be non-uniformly decimated [AS96, RR96, XV96]. Another way to achieve non-uniform decimation is by using simplification envelopes [CVM+96]. An envelope consists of an inner and outer surface, and the simplification is constrained to fall between these two surfaces. Volumetric methods can also be employed by voxelizing the original mesh into a multi-resolution hierarchy, and extracting iso-surfaces of varying complexity [HHVW96, SFYC96].

A very different approach to providing many different levels of detail from a complex model is a theoretically sound framework for multi-resolution models [EDD+95, CPD+96]. A multi-resolution model consists of a simple base mesh, which is triangulated, and a series of local wavelet coefficents, which capture the details of the original mesh at various resolutions.

2.3 Volume Decimation

The surface methods are designed to reproduce the geometry of the surface as faithfully as possible. For volume rendering, it is not necessary to preserve the geometry of the volume, since the geometry is not readily perceived in the final image. However, we still wish to produce an image which is as close to the original as possible using fewer cells. Since the running time of most visualization techniques is dependent on the number of cells in the volume, this translates directly to improved performance.

Cignoni *et al.* have presented a method for deriving multiple resolutions of a scattered volume data set [CDFM+94]. The scattered data is visualized by first tetrahedralizing it, and then applying standard visualization techniques for tetrahedral data sets. The multi-resolution model is built by constructing a

series of tetrahedralizations based on decreasing tolerance values. The first tetrahedralization is built using a small subset of the vertices, which permits approximation of all the other vertices within a certain error bound. The next tetrahedralization is built from the previous by adding vertices until the new error criterion is met. A disadvantage of this method is the need to re-grid the volume by adding edges not present in the original volume.

Of course, instead of decimating the volume before it is rendered, some rendering techniques intrinsically allow for multiple levels of detail. Volume rendering algorithms which are slice-based allow the user to define the number of slices used (for instance [YRL+96]) and can be seen as a limited form of multi-resolution rendering. We can also achieve multiple levels of detail by modeling the data itself in a hierarchical fashion [LH91, WVG94, WVGTG96]. Hierarchical data models allow standard volume rendering methods to work on volumes of different sizes. Building hierarchies is most effective for regular grids, since the hierarchical decomposition can easily follow cell boundaries. However, the rendered images typically have artifacts due to the discontinuities which arise because different parts of the volume are rendered at different levels of hierarchy. Using hierarchies also adds a significant memory overhead.

3 Polygon-Based Volume Rendering and Decimation

For rendering decimated volumes, we use a direct volume rendering system which is based on a generalized software scan conversion of *polygons*, rather than the more conventional ray-casting, projection, or splatting of *cells* [WVGTG96]. This difference is motivated by the fact that any cell type can be decomposed into a set of bounding polygons. This method also solves two key problems which are important to this work. First, it is able to render overlapping cells which occur in multi-grid data sets. Second, it is able to handle grids which are not made up of simple cell types. The scan conversion algorithm generalizes traditional polygon scan-line methods in that it renders semi-transparent regions of space between polygons, as well as opaque surfaces. This method requires no graphics hardware, and produces excellent quality images. For more details on the rendering algorithm, and comparisons to other methods, see [WVGTG96]. The rendering time is based on the number of polygons in the volume, after culling out those which don't lie in the viewing frustum and those smaller than one pixel. To achieve faster rendering times, we need to further reduce the number of polygons used to render the image.

Because the renderer sees only polygons, it is completely independent of grid type. We exploit this advantage when building simplified models. The decimation process is not constrained to produce only simple hexahedral or tetrahedral cells. This is important since, when decimating polygons, we cannot afford to re-grid the volume to maintain simple cell types. Thus, the decimation process turns a grid made of all hexahedra or tetrahedra into a hybrid grid with many-sided cells, and some polygons which are not a part of any cell. Given a volume of polygons, we identify those that are not needed to render the volume. For instance, if all the polygons in a sub-volume map to the exact same color, then clearly the internal polygons are not needed for rendering. We only need the polygons that form the border (outer surface) of that sub-volume.

3.1 The Decimation Process

The basic decimation process proceeds as follows. We examine the whole volume at the vertex level, attempting to identify important vertices. Un-important vertices can be decimated. If a vertex is decimated, all polygons which include this vertex are decimated as well. Thus, vertex decimation is equivalent to polygon decimation, in our algorithm.

Our initial attempt at this decimation process was strictly local. We examined the region comprised of a vertex and its neighbors. We were able to assign an error metric based on the linearity of this region. If the local region was very linear in all directions, then the region could be represented without the interior vertex, or the polygons associated with it. Indeed, this method did work well locally, but globally it led to many problems. The worst problem was the appearance of holes in images of the volume. A hole appeared where a long string of vertices along the view were all decimated due to their local linearity. There were also bad artifacts when strong non-linearities in the transfer function (particularly around the free-stream in CFD volumes) caused certain small areas to allow their color to bleed very noticeably into the otherwise clear portion of the volume. Figure 2 shows an example of these problems.

To solve these problems, we propose a decimation method based on dividing up the data range into several sub-ranges. There are certain boundaries in the volume which must be maintained for an intelligent visualization. In a CFD volume, the boundary between the free-stream and other data values is one of these. In medical data, a few examples are the boundary between bone, different types of tissue, and air. Our decimation method preserves these boundaries as accurately as possible, and only decimates the regions between them.

The first step in the decimation process is to break up the data range into several sub-ranges, or buckets. Each vertex is then mapped into one bucket. The bucket boundaries represent the critical values in the data which contribute to the boundaries of interesting features in the rendered image. Hence adjacent vertices which lie on opposite sides of these bucket boundaries are very important. The other vertices can be decimated from the volume without losing too much information. The bucket boundaries can be placed in several ways, and we will discuss several methods in the next section.

The next step is to traverse the volume examining each vertex and its neighbors. In a regular or curvilinear dataset several notions of "neighbor" are possible. For this work we consider each vertex to have 26 neighbors found by incrementing the (i, j, k) location of the vertex by ± 1 in each direction. For tetrahedral grids, an explicit list of neighbors must be generated for each vertex.

If the vertex maps to the same bucket as all of its neighbors, then it will be decimated. Since buckets are placed to indicate regions of related data values, and which map to similar colors, our heuristic is to throw the vertex away hoping that the neighbors will do a sufficient job of representing the region.

After the decimation pass has occurred and all vertices have been examined, each decimated region consists of a set of connected vertices that map to the same bucket, none of which is adjacent to a vertex in a different bucket. The only vertices left active in the volume are those which are adjacent to vertices in buckets different from the one they are in. These vertices should represent the important boundaries in the volume. We also ensure that the vertices on the exterior boundary of any grid are retained. These are necessary to identify the region of space which is inside the volume.

In our implementation, we simply keep a boolean array indicating which polygons are in the volume. To reduce even this memory overhead, the polygon list could be sorted so the decimated vertices are listed first, and an offset could be kept to indicate where the active, or un-decimated, vertices begin.

3.2 Bucket Placement

We explored five automatic strategies for bucket placement: *uniform by range*, *uniform by histogram*, *histogram curvature*, *histogram features*, and *transfer function*. It is also possible for the user to guide the placement of the buckets interactively. Raw histograms consist of 256 equal-width buckets spanning the range of the data. Figure 1 shows raw histograms for three different CFD volumes. However, all methods that use "histogram" actually use a histogram that has been smoothed by a Gaussian filter of standard deviation 2.82.

1. **Uniform by Range**: The simplest strategy places a user-specified number of buckets at equal intervals over the data range of the volume. More buckets typically lead to less decimation. This method sometimes works well, but often generates artifacts due to imprecise bucket placement. Also, when the user changes the number of buckets by one, all but the first and last buckets move. Since the decimation is very sensitive to bucket boundaries, moving the buckets in this manner can lead to un-intuitive changes in the decimation of the volume.

2. **Uniform by Histogram**: Our second strategy is equally simple. Instead of placing buckets uniformly, based on the range of data, we place them uniformly based on the frequency of data, as judged by the smoothed histogram. Smoothing the histogram lowers extreme peaks (such as the free stream of aeronautical simulations), and allows the buckets to be spread more evenly.

   If based on the *raw* histogram, the approximately same number of vertices are found in each bucket. It is very hard to get substantial decimation rates using this strategy, since many buckets tend to get placed where there is a lot of data. While this protects important data ranges from being decimated, it also prevents most of the vertices from being decimated. Inferior performance on raw histograms motivated the use of smoothing.

3. **Histogram Curvature**: This strategy places the bucket boundaries based on the magnitudes of the second derivatives of the smoothed histogram. Bucket boundaries are thus placed in regions of high curvature. A typical histogram will have several flat areas, separated by steep spikes or valleys. These spikes and valleys often represent interesting features in the volume. We would like to place a bucket boundary on each side of such spikes and valleys to preserve the boundaries between this feature and other parts of the volume. If h_i represents the histogram value at i, this method places buckets where $|(h_{i-1} - 2h_i + h_{i+1})|$ is largest. The number of buckets can be chosen by the user. This strategy has fewer artifacts than the previous two strategies, but it often places more buckets than are ideally necessary. In particular, it often places boundaries not only on both sides of spikes and valleys, but along these features as well. Again, smoothing the histogram can eliminate noise which can produce high curvatures in the histogram, but which doesn't really represent a feature in the data.

4. **Histogram Feature**: This strategy analyzes the smoothed histogram and attempts to find the features discussed above. Once these features are found, a good strategy would be to place a bucket boundary on either side, and avoid placing a bucket in the middle of the feature. The strategy proceeds as follows. First, the histogram is scanned to identify the point with the highest curvature magnitude. This is the center of a feature. We then scan both left and right, looking for the next peak in the curvature of opposite sign, on opposite sides of the feature. A bucket boundary is placed at both of these data values. We then begin the search again, excluding any features already found, or any histogram entries which already have bucket boundaries. This can proceed to place as many buckets as the user likes.

5. **Transfer Function**: This strategy addresses the problem with previous strategies that they do not take the transfer function into account. The boundaries that are perceived in the final images are largely the result of the transfer function, and not simply the underlying data. Transfer functions are often used to produce sharp changes in hue, even when the data is smoothly varying, to perceive surfaces and subtle variations in the data. Our rendering system uses a piecewise linear transfer function. A control point in the transfer function usually indicates a change in the hue at the corresponding data value. We can generate a very good set of buckets for the decimation process by simply placing a bucket boundary at each control point. Figure 3 shows a typical histogram and transfer function, and illustrates where the buckets would be placed. The transfer function also often has more detail in

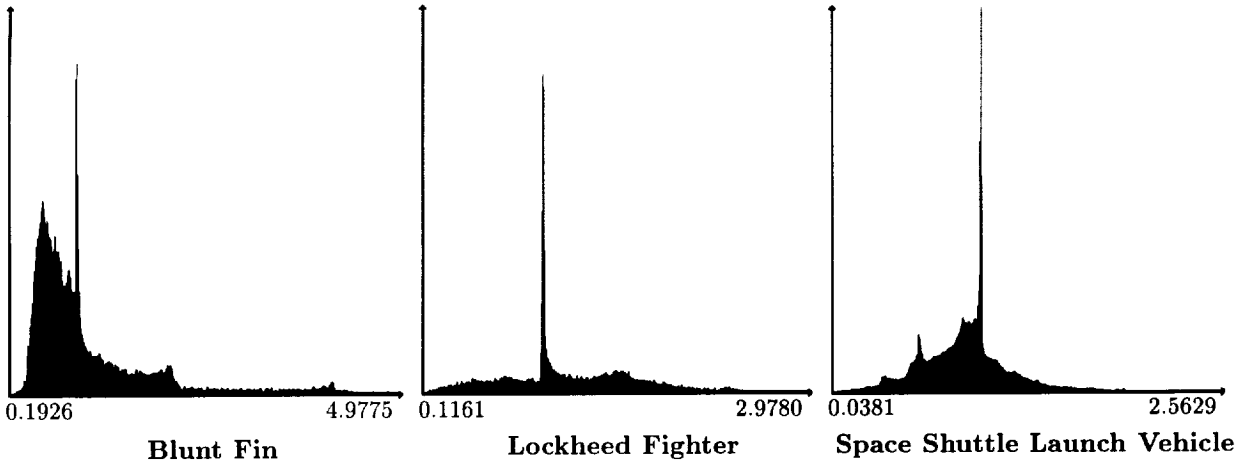

Figure 1: Histograms of three CFD volumes.

regions of high interest, so this method places more buckets in these regions as well. This method of generating buckets can produce images which are practically identical to images rendered using the non-decimated volume.

The user can often increase the percentage of polygons decimated by using a subset of the bucket boundaries generated automatically. This is particularly true of the *histogram feature* and the *transfer function* strategies. If the user knows what data values correspond to important features, he can help the histogram feature method by choosing which features to use. When using the transfer function strategy, it is often wise to reduce the complexity of the transfer function, so fewer buckets are used. It should be noted, however, that reducing the number of bucket boundaries may increase the number of artifacts that appear in the final images.

4 Image-Based Error Metric

While volume rendering does not display the geometry of the volume, it does display the *light field* associated with it. If we are to decimate volumes successfully for direct volume rendering, we must be able to analyze the effect of decimation on the light field produced by the volume.

The notion of a light field was introduced as an image-based rendering architecture [LH96, GGSC96]. Levoy and Hanrahan defined an object's light field as a 4D function of position and direction in regions of free space. This function describes the light emitted by the object. A single rendered image is a 2D slice of this 4D function. Therefore, an appropriate sampling of rendered images can be used as an approximation of this light field. To give us a metric on how the decimation process has effected the visualization of the dataset, we want to compare the light field produced by the decimated volume with the light field produced by the original volume.

We represent the light field with 32 "x-ray" images, two per viewpoint, taken at regular spacing on the hemisphere surrounding one side the volume. Image resolution was 500 by 500 pixels. The x-ray images are generated using a linear gray-scale transfer function with zero opacity. Thus the image from the spherically opposing viewpoint is identical, and need not be computed. The absence of opacity allows us to integrate the density (or other field function) through the volume, producing an approximate 3D analog of the Radon transformation [Mal93, CCF94, LH96].

Method	*Blunt Fin*	*Lockheed Fighter*	*Space Shuttle*
Uniform by Range	0.089	0.056	0.061
Uniform by Histogram	0.117	0.042	0.040
Histogram Curvature	0.062	0.048	0.004
Histogram Features	0.058	0.065	0.031
Transfer Function	0.074	0.071	0.022

Table 1: Errors computed using the image-based error metric, as discussed in section 4. All data sets were decimated approximately 45–62%.

For our CFD data, we had to generate two images per viewpoint using separate linear gray-scale transfer functions: one to show the data below free-stream, and the other to show the data above free-stream. If a single gray-scale ramp were used, the free-stream data would obliterate any interesting features.

The sixteen views are generated by tessellating a sphere into 60 congruent triangles [VGK96], and using the vertices of the triangles as the viewing directions. A ray from each vertex through the center of the sphere indicates a viewing direction. This method considers both an icosahedron, which has twelve vertices, and its dual figure, the dodecahedron, which has 20 vertices, for a total of 32 vertices. Figure 4 shows the vertices of these dual polyhedra, and figure 5 shows the 16 views of the data above free stream in the *blunt fin* (section 5).

To evaluate the error of decimation in one direction, the decimated version of the image is compared with the undecimated version, and pixel by pixel differences (with gray level scaled from 0 to 1.0) are computed. Since the two images were generated from precisely the same view, using the same rendering software, we do not need to worry about registering the images. The sum-of-squares of these differences is accumulated and averaged. For averaging purposes, pixels that were black in both images of the pair are discarded (i.e., do not contribute to the number of "observations").

After comparing the 32 image pairs, we have a set of 32 error values. In this paper we present the square root of the average of these values, which corresponds to the standard deviation, as a global error measure. This treatment of the errors is preliminary, and requires further study. Possible alternatives would weight larger errors more heavily, with the extreme example being to use the maximum pixel error as the measure for the whole dataset.

5 Experimental Results

We examined the results of the decimation algorithm on three volumes, and compared the images generated by several methods. We used the following CFD volumes: the *blunt fin* [HB85], a single curvilinear grid of 40,960 data points; the *Space Shuttle launch vehicle* [BCFM+89], consisting of nine curvilinear grids with 941,159 data points; and the *Lockheed fighter* (courtesy of John Batina of NASA Langley Research Center), an unstructured tetrahedral grid consisting of 13,382 data points and 70,125 tetrahedra. Comparable results were observed on CT data, but are not described further.

Table 1 shows the error generated using different bucket methods (see section 4). All methods were adjusted to produced volumes that were about 45–62% decimated. No method is the winner in all cases, and many of the errors are closely clustered. Errors over 0.10 seem to predict substantial artifacts in the images. In particular, the decimation method used for Figure 2, which was discussed in section 3.1, has an error value of 0.203. In practice we have found the uniform by histogram and histogram curvature methods

	Blunt Fin	*Lockheed Fighter*	*Space Shuttle*
No Decimation:			
polygons total	173,752	290,096	1,524,791
CPU seconds	73.2	228.6	328.8
Transfer Function strategy:			
polygons retained	68,029	160,358	726,969
decimation	61%	44%	52%
CPU seconds	26.5	101.6	148.6
speedup	2.8	2.3	2.2
Histogram Feature strategy:			
polygons retained	67,956	182,559	550,764
decimation	61%	37%	64%
CPU seconds	25.4	116.6	125.3
speedup	2.9	2.0	2.6

Table 2: This table shows the decimation and rendering speedup obtained using the two more successful decimation methods, *transfer function* and *histogram features*, on three CFD volumes. The images corresponding to these numbers are figures 6, 8, and 10. All times are based on a 150-MHz R4400 processor. Polygon counts refer to polygons processed for these images, not those in the whole volume.

hard to control. However, the histogram features method is intuitive and easy to control, so it was selected as the data-based method for image production.

The image-based error metric gives us an indication of how well the decimated volume represents the underlying data. However, the transfer function strategy makes its decimation choices based on a certain transfer function. These choices may not reflect the underlying data as well as other methods, yet still give superior results when the data is visualized *with the same transfer function.* For this reason, the transfer function is capable of producing actual color images with much less error than the image-based error metric suggests.

Figures 6 – 11 show decimated and un-decimated images using the histogram features and the transfer function strategies. Table 2 shows the decimation and speedup obtained for figures 6, 8, and 10. The decimated volumes can generally be drawn 2 to 3 times faster than un-decimated volumes, and generally are very close to the original images. Figures 7, 9, and 11 provide closeups of interesting regions in the volumes. The decimation rate for the closeup images in nearly identical to the decimation rate in the entire volume, so the decimation algorithms are removing polygons fairly equally from the entire volume.

As expected, the blunt fin proved to be an easy volume to decimate. We obtained decimation rates of over 60 percent with almost no artifacts visible. The images of the Lockheed fighter also have few artifacts, but the decimation rates are lower, close to 40 percent. This may be because the fighter volume has many more complex boundaries than the blunt fin. The small white dots on the images of the decimated fighter are the fighter's opaque surface showing through the volume, and hiding volume polygons. The fighter's surface geometry was not considered in the decimation calculation.

The Space Shuttle proved to be a difficult image to decimate without artifacts, due to the very large and very thin cone of air just slightly above free stream, which surrounds the shuttle. The histogram curvature strategy (no image shown) gave unusually good results by focusing exclusively on the border between the free-stream and neighboring data values, and ignoring other features (see table 1). As one can see in figure 10,

both the histogram features and transfer function strategies produce some artifacts on the large green surface. However, all the important features in the volume are still readily visible, and the images still carry the same information as the un-decimated image.

The decimation algorithm took 0.85 seconds for the blunt fin, 4.26 seconds for the fighter, and 26.6 seconds for the space shuttle datasets, using a 150-MHz R4400 processor. As can be seen from table 2, these times are only a fraction of the time it takes to render the final images.

6 Conclusion and Future Work

We need to better understand the elements in a volume that play an *active* role in the final image. Only a stronger understanding of these roles will help us produce substantially faster, perhaps real-time, volume rendering architectures for large irregular grids.

This work has shown that one can indeed remove over 50 percent of the polygons in a volume, and not readily tell the difference in the final images. However, the decimation process is very sensitive to the preservation of important boundary surfaces in the volume. It is hard to get decimation rates much larger than 50 percent by our methods, and still preserve image quality, because important surfaces tend to get violated. We have shown that by identifying the key changes in color, we can remove those portions of the volume which lie in between the transition points without substantial loss in the image quality in several data sets. However, more study is needed before we can tell whether this is a general phenomenon.

7 Acknowledgments

Funds for the support of this study have been allocated by NAS/NASA Ames Research Center, Moffett Field, California, Grant Number NAG 2-991, and by the National Science Foundation, Grant Number CCR 9503829.

References

[AS96] Maria-Elena Algorri and Francis Schmitt. Mesh simplification. *Computer Graphics Forum (EUROGRAPHICS '96)*, 15(3):C–77–C–86, 1996.

[BCFM+89] P.G. Buning, I.T. Chiu, Jr. F.W. Martin, R.L. Meakin, S. Obayashi, Y.M. Rizk, J.L. Steger, and M. Yarrow. Flowfield simulation of the space shuttle vehicle in ascent. *Fourth International Conference on Supercomputing*, 2:20–28, 1989.

[CCF94] Brian Cabral, Nancy Cam, and Jim Foran. Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. In *1994 Symposium on Volume Visualization*, pages 91–98, Washington, D.C., October 1994. ACM.

[CCMS96] A. Ciampalini, P. Cignoni, C. Montani, and R. Scopigno. Multiresolution decimation based on global error. *submitted for publication*, 1996.

[CDFM+94] Paolo Cignoni, Leila De Floriani, Claudio Montani, Enrico Puppo, and Roberto Scopigno. Multiresolution modeling and visualization of volume data based on simplicial complexes. In *1994 Symposium on Volume Visualization*, Washington, D.C., October 1994. ACM.

[CVM+96] Jonathan Cohen, Amitabh Varshney, Dinesh Manocha, Greg Turk, Hans Weber, Pankaj Agarwal, Frederick P. Jr. Brooks, and William Wright. Simplification envelopes. In *Proceedings of SIGGRAPH*, Computer Graphics, Annual Conference Series, pages 119–128, August 1996.

[DZ91] Michael J. DeHaemer, Jr. and Michael J. Zyda. Simplification of objects rendered by polygonal approximations. *Computers and Graphics*, 15(2):175–184, 1991.

[EDD+95] Matthias Eck, Tony DeRose, Tom Duchamp, Hugues Hoppe, Michael Lounsberg, and Werner Stuetzle. Multiresolution analysis of arbitrary meshes. In *Proceedings of SIGGRAPH*, Computer Graphics, Annual Conference Series, pages 173–182, August 1995.

[FDFH90] James D. Foley, Andies Van Dam, Steven Feiner, and John Hughes. *Computer Graphics: Principles and Practice.* Addison-Wesley Publishing Company, Reading, Mass., 2nd edition, 1990.

[Gar90] Michael P. Garrity. Raytracing irregular volume data. *Computer Graphics*, 24(5):35–40, December 1990.

[GGSC96] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. In *Proceedings of SIGGRAPH*, Computer Graphics, Annual Conference Series, pages 43–54, August 1996.

[GH95] Michael Garland and Paul S. Heckbert. Fast polygonal approximation of terrains and height fields. Technical Report CMU-CS-95-181, Carnegie Mellon University, School of Computer Science, Carnegie Mellon University, Pittsburg, PA 15213, 1995.

[Gie92] Christopher Giertsen. Volume visualization of sparse irregular meshes. *IEEE Computer Graphics and Applications*, 12(2):40–48, March 1992.

[GP93] Christopher Giertsen and Johnny Peterson. Parallel volume rendering on a network of workstations. *IEEE Computer Graphics and Applications*, pages 16–23, November 1993.

[HB85] Ching-Mao Hung and Pieter G. Buning. Simulation of blunt-fin-induced shock-wave and turbulent boundary-layer interaction. *J. Fluid Mechanics*, 154:163–185, 1985.

[HDD+93] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Mesh optimization. *Proceedings of SIGGRAPH*, 25(2):19–25, August 1993.

[HH93] Paul Hinker and Charles Hansen. Geometric optimization. In Gregory Nielson and Dan Bergeron, editors, *Visualization '93*, pages 189–195, San Jose, Ca, October 1993. IEEE.

[HHVW96] Taosong He, Lichan Hong, Amitabh Varshney, and Sidney W. Wang. Controlled topology simplification. *IEEE Transactions on Visualization and Computer Graphics*, 2(2):171–183, June 1996.

[Hop96] Hugues Hoppe. Progressive meshes. In *Proceedings of SIGGRAPH*, Computer Graphics, Annual Conference Series, pages 99–108, August 1996.

[KLS96] Reinhard Klein, Gunther Liebich, and Wolfgang Strasser. Mesh reduction with error control. In *Visualization '96*, pages 311–318. IEEE, October 1996.

[Koy92] Koji Koyamada. Fast traversal of irregular volumes. In T. L. Kunii, editor, *Visual Computing - Integrating Computer Graphics and Computer Vision*, pages 295–312. Springer Verlag, 1992.

[KT96] Alan D. Kalvin and Russ H. Taylor. Superfaces: Polyhedral approximations with bounded error. *IEEE Computer Graphics and Applications*, 16(3):64–77, May 1996. (An erratum with correct figures appears in the July 1996 issue).

[Lev92] Marc Levoy. Volume rendering using the fourier projection-slice theorem. In *Proceedings of Graphics Interface '92*, Vancouver, B.C., 1992. Also Stanford University Technical Report CSL-TR-92-521.

[LH91] David Laur and Pat Hanrahan. Hierarchical splatting: A progressive refinement algorithm for volume rendering. *Computer Graphics (ACM Siggraph Proceedings)*, 25(4):285–288, July 1991.

[LH96] Marc Levoy and Pat Hanrahan. Light field rendering. In *Proceedings of SIGGRAPH*, Computer Graphics, Annual Conference Series, pages 31–42, August 1996.

[LKR+96] Peter Lindstrom, David Koller, William Ribarsky, Larry F. Hodges, Nick Faust, and Gregory A. Turner. Real-time, continuous level of detail rendering of height fields. In *Proceedings of SIGGRAPH*, Computer Graphics, Annual Conference Series, pages 109–118, August 1996.

[LL94] Philippe Lacroute and Marc Levoy. Fast volume rendering using a shear-warp factorization of the viewing transformation. In *Proceedings of SIGGRAPH*, Computer Graphics, Annual Conference Series, pages 451–458, July 1994.

[Luc92] Bruce Lucas. A scientific visualization renderer. In *Visualization '92*, pages 227–233. IEEE, October 1992.

[Ma95] Kwan-Liu Ma. Parallel volume ray-casting for unstructured grid data on distributed memory architectures. In *1995 Parallel Rendering Symposium*, pages 23–30. ACM, 1995.

[Mal93] Tom Malzbender. Fourier volume rendering. *ACM Transactions on Graphics*, 12(3):233–250, July 1993.

[MHC90] Nelson Max, Pat Hanrahan, and Roger Crawfis. Area and volume coherence for efficient visualization of 3d scalar functions. *Computer Graphics (ACM Workshop on Volume Visualization)*, 24(5):27–33, December 1990.

[MHK95] Xiaoyang Mao, Lichan Hong, and A. Kaufman. Splatting of curvilinear volumes. In *Visualization '95*, pages 61–68. IEEE, November 1995.

[RB93] J. Rossignac and P. Borrel. Multi-resolution 3D approximations for rendering complex scenes. In B. Falcidieno and T. L. Kunii, editors, *Geometric Modeling in Computer Graphics*, pages 455–465. Springer Verlag, 1993.

[RR96] Remi Ronfard and Jarek Rossignac. Full-range approximation of triangulated polyhedra. *Computer Graphics Forum (EUROGRAPHICS '96)*, 15(3), 1996.

[SFYC96] Raj Shekhar, Elias Fayyad, Roni Yagel, and J. Fredrick Cornhill. Octree-based decimation of marchin cubes surfaces. In *Visualization '96*, pages 335–344. IEEE, October 1996.

[SMK96] Cláudio Silva, Joseph S. B. Mitchell, and Arie E. Kaufman. Fast rendering of irregular grids. In *1996 Symposium on Volume Visualization*, pages 15–22. ACM, October 1996.

[ST90] Peter Shirley and Allan Tuchman. A polygonal approximation to direct scalar volume rendering. *Computer Graphics*, 24(5):63–70, December 1990.

[SZL92] Williams J. Schroeder, Jonathan A. Zarge, and William E. Lorensen. Decimation of triangle meshes. *Computer Graphics (ACM Siggraph Proceedings)*, 24(2):65–70, July 1992.

[Tur92] Greg Turk. Re-tiling polygonal surfaces. In *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 55–64, July 1992.

[Use93] Sam Uselton. Parallelizing VOLVIS for multiprocessor SGI workstations. Technical Report RNR-93-013, NAS-NASA Ames Research Center, Moffett Field, CA, 1993.

[VGK96] Allen Van Gelder and Kwansik Kim. Direct volume rendering with shading via three-dimensional textures. In *1996 Symposium on Volume Visualization*, pages 23–30. ACM, October 1996.

[VGW93] Allen Van Gelder and Jane Wilhelms. Rapid exploration of curvilinear grids using direct volume rendering. In *Visualization '93*, San Jose, CA, October 1993. IEEE. (extended abstract) Also, University of California technical report UCSC-CRL-93-02.

[Wil92] Peter Williams. Interactive splatting of nonrectilinear volumes. In *Visualization '92*, pages 37–44. IEEE, October 1992.

[WVG94] Jane Wilhelms and Allen Van Gelder. Multi-dimensional trees for controlled volume rendering and compression. In *ACM Symposium on Volume Visualization 1994*, Washington, D.C., October 1994. See also technical report UCSC-CRL-94-02.

[WVGTG96] Jane Wilhelms, Allen Van Gelder, Paul Tarantino, and Jonathon Gibbs. Parallel hierarchical direct volume rendering for irregular and multiple grids. In *IEEE Visualization '96 Conference Proceedings*, October 1996.

[XV96] Julie C. Xia and Amitabh Varshney. Dynamic view-dependent simplification for polygonal models. In *Visualization '96*, pages 327–334. IEEE, October 1996.

[YRL+96] Roni Yagel, David M. Reed, Asish Law, Po-Wen Shih, and Naeem Shareef. Hardware assisted volume rendering of unstructured grids by incremental slicing. In *1996 Symposium on Volume Visualization*, pages 55–62. ACM, October 1996.

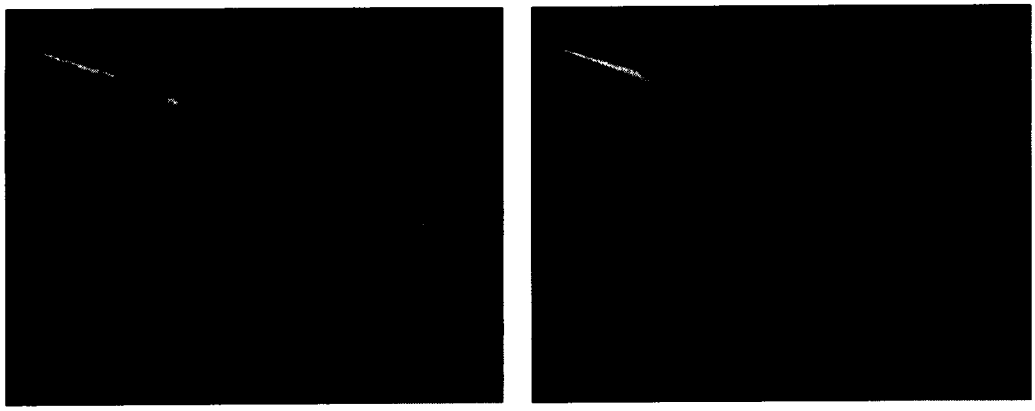

Figure 2: This figure illustrates several problems with our original error metric, which did not preserve the important boundaries in the data. The image on the left is a decimated version of the image on the right. The decimated image is comprised of approximately 140,000 polygons, while the un-decimated image has approximately 280,000 polygons.

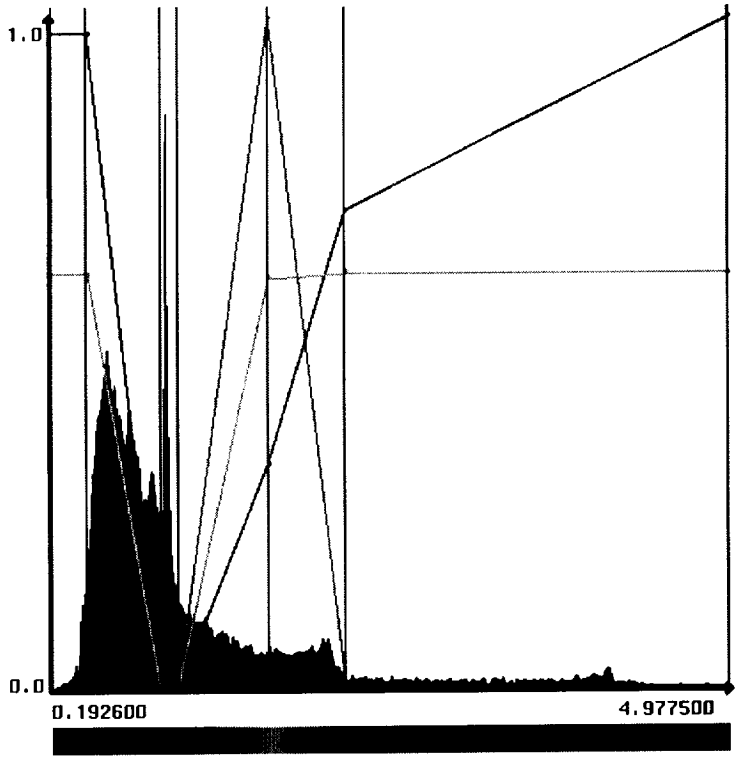

Figure 3: This figure show a typical transfer function. The red, green, blue and cyan lines represent the red, green, blue and opacity portions of the transfer function. The vertical magenta lines illustrate the bucket boundaries which were obtained from the transfer function. The histogram is shown in gray in the background.

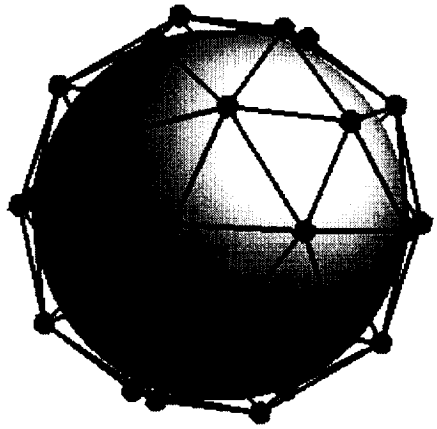

Figure 4: This figure shows a sphere tessellated into 32 viewing directions. The light vertices are from an icosahedron and the dark vertices are from its dual dodecahedron.

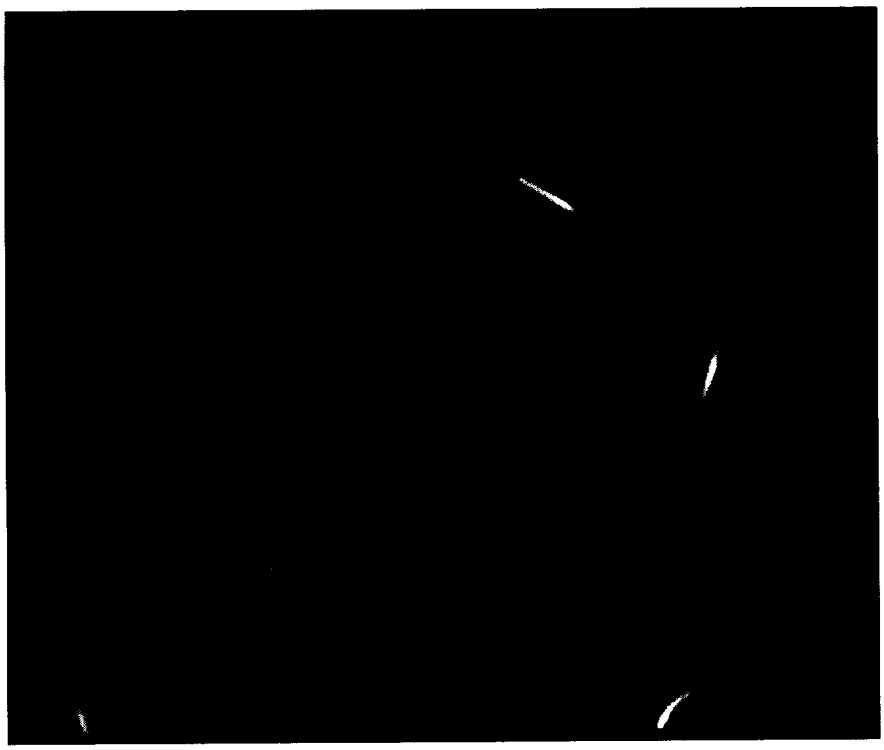

Figure 5: This figure show the 16 (condensed) views of the data above free stream in the *blunt fin*. These are the views used by the image-based error metric.

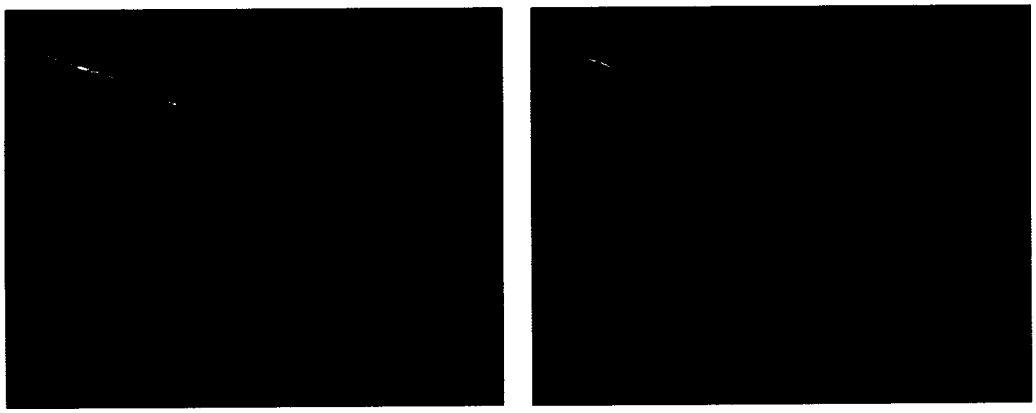

Figure 2: This figure illustrates several problems with our original error metric, which did not preserve the important boundaries in the data. The image on the left is a decimated version of the image on the right. The decimated image is comprised of approximately 140,000 polygons, while the un-decimated image has approximately 280,000 polygons.

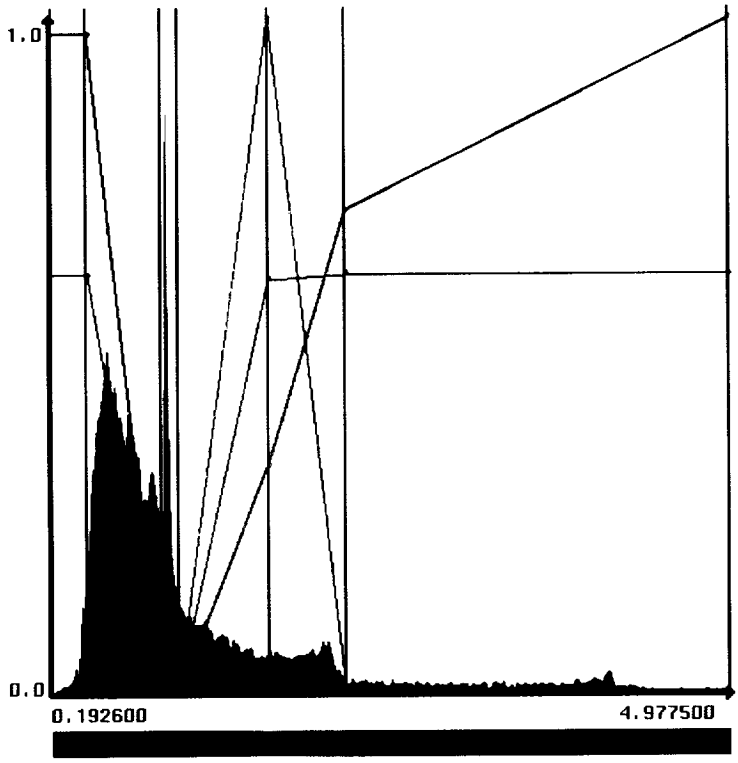

Figure 3: This figure show a typical transfer function. The red, green, blue and cyan lines represent the red, green, blue and opacity portions of the transfer function. The vertical magenta lines illustrate the bucket boundaries which were obtained from the transfer function. The histogram is shown in gray in the background.

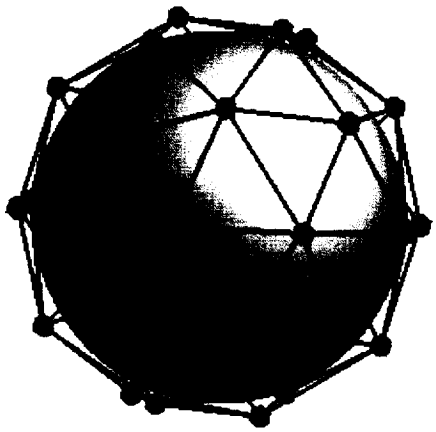

Figure 4: This figure shows a sphere tessellated into 32 viewing directions. The light vertices are from an icosahedron and the dark vertices are from its dual dodecahedron.

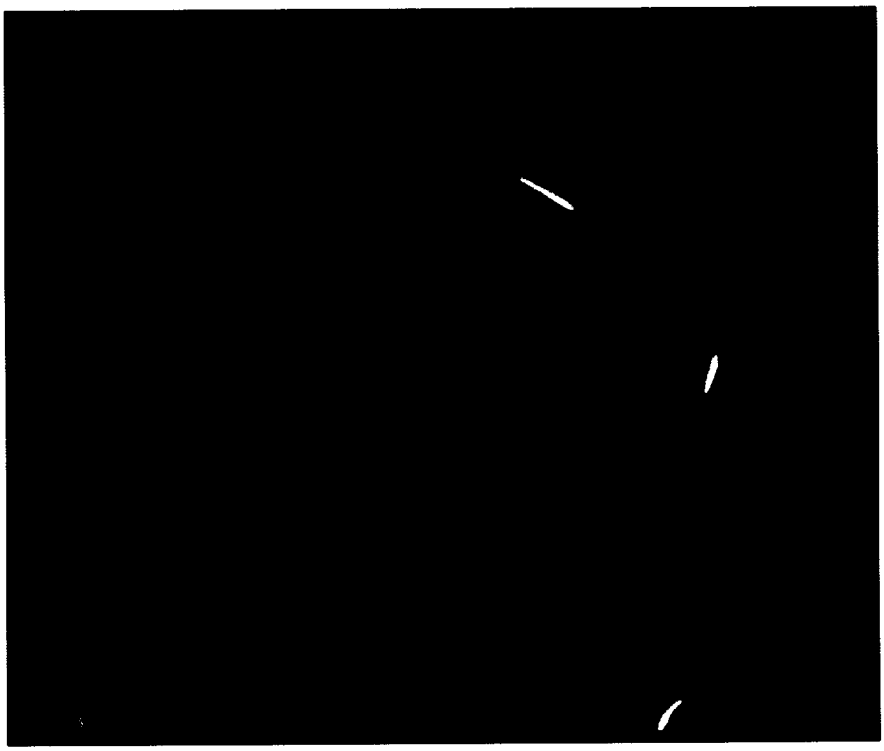

Figure 5: This figure show the 16 (condensed) views of the data above free stream in the *blunt fin*. These are the views used by the image-based error metric.

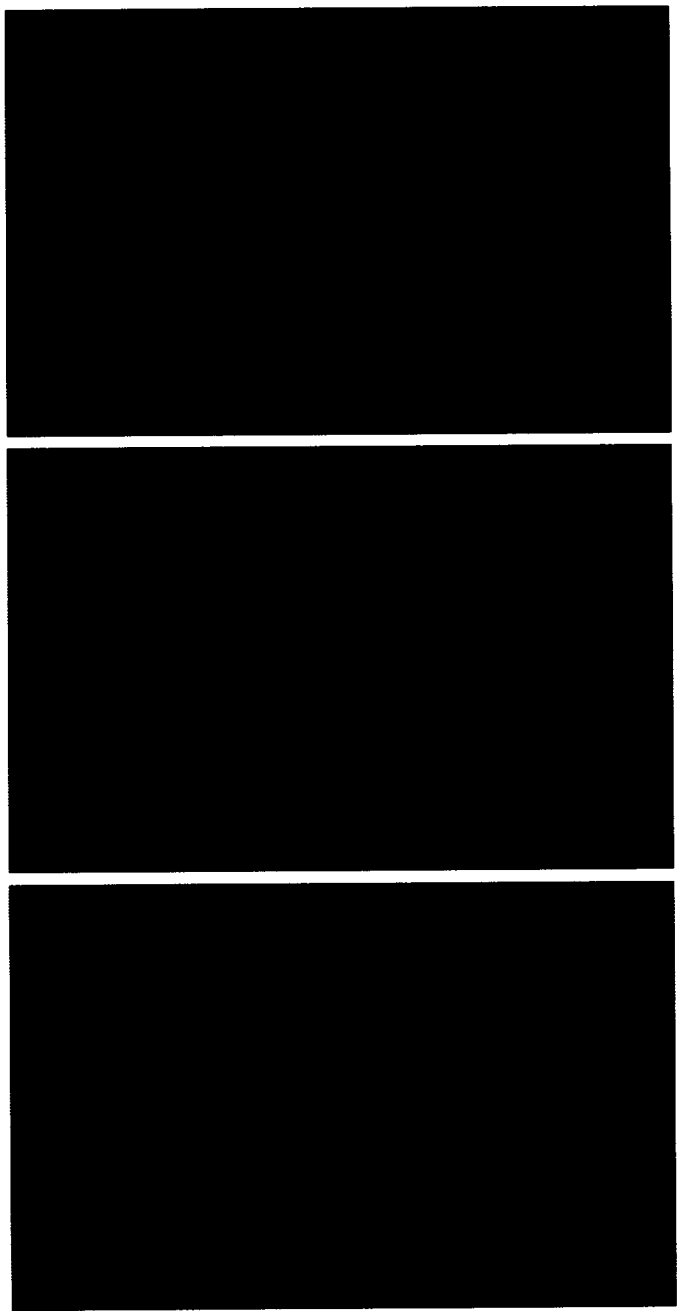

Figure 6: This figure shows the *blunt fin* volume, both with and without decimation. The top image is the un-decimated volume. The middle image is decimated using the transfer function strategy, obtaining a decimation of 61%, and a speedup of 2.8. The bottom image is decimated using the histogram feature strategy, and obtained a 61% decimation and a 2.9 times speedup.

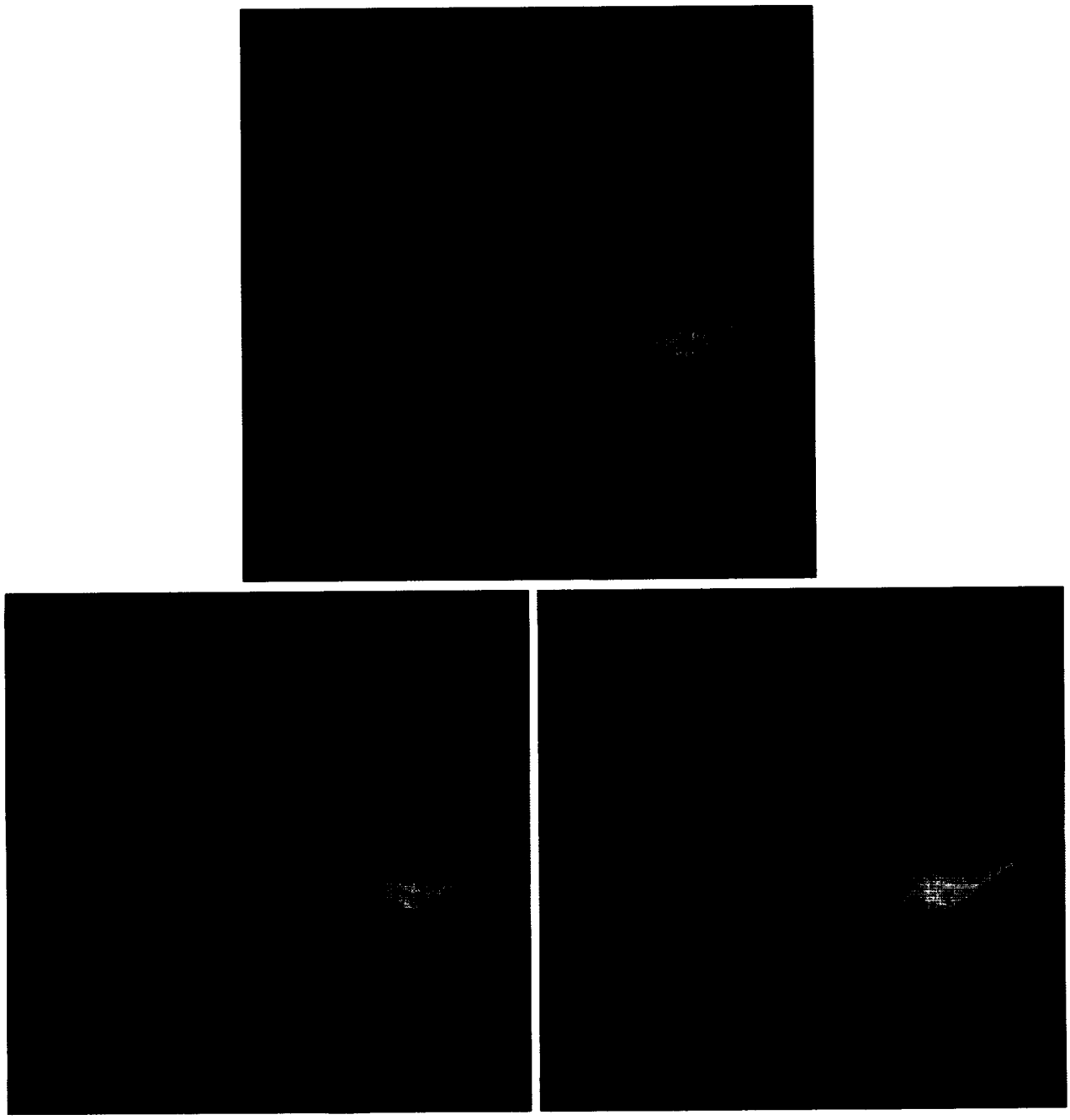

Figure 7: This figure shows a closeup of *blunt fin* volume. The top volume is not decimated, and the bottom two were generated using the same decimation as in figure 6.

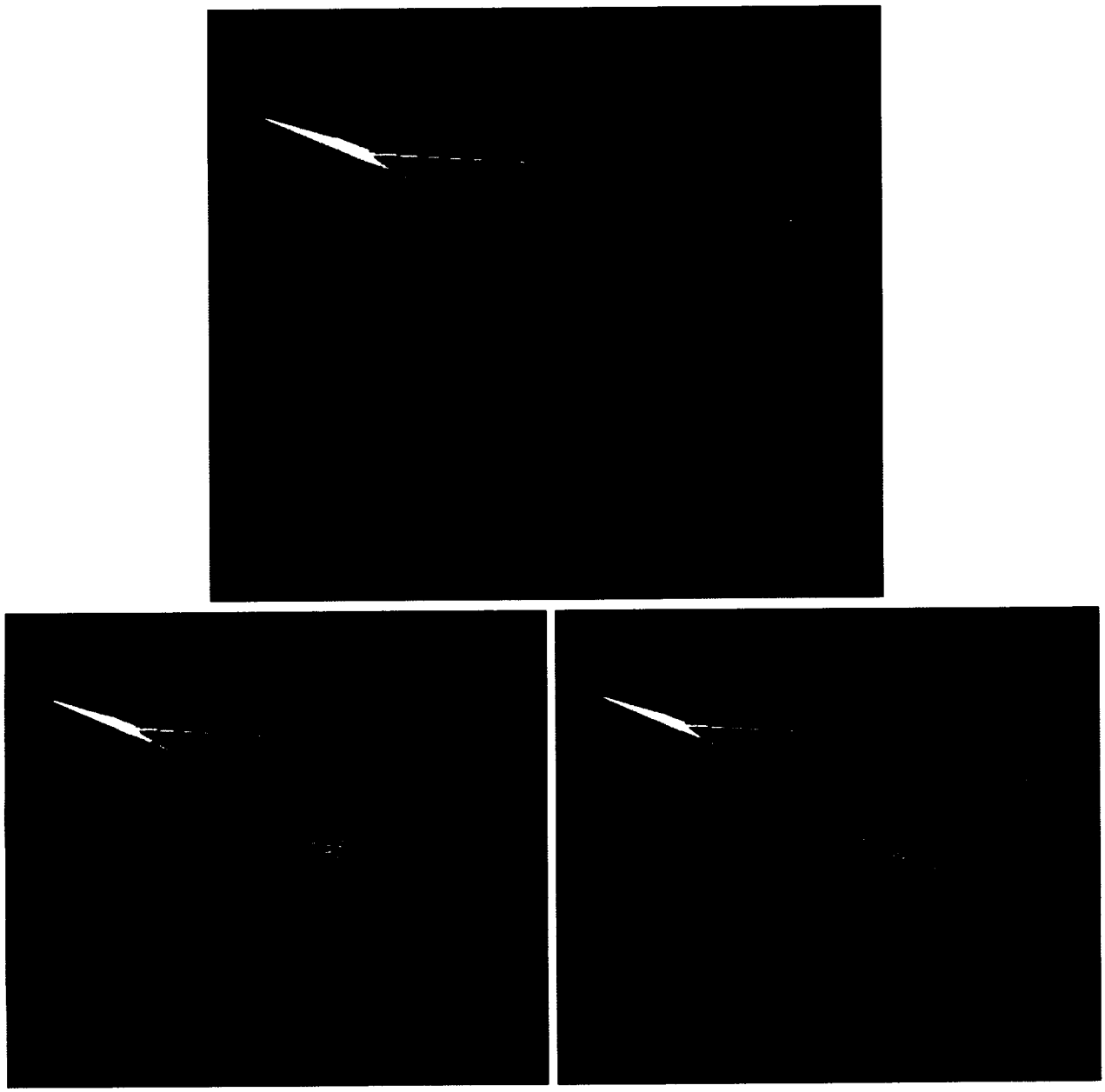

Figure 8: This figure shows the *Lockheed fighter* volume. The top image is without decimation. The transfer function strategy was used on the bottom-left image to obtain a speedup of 2.3 times and a decimation of 44%. The bottom-right image used the histogram feature strategy and obtained a speedup of only 2 times, and a decimation of only 37%.

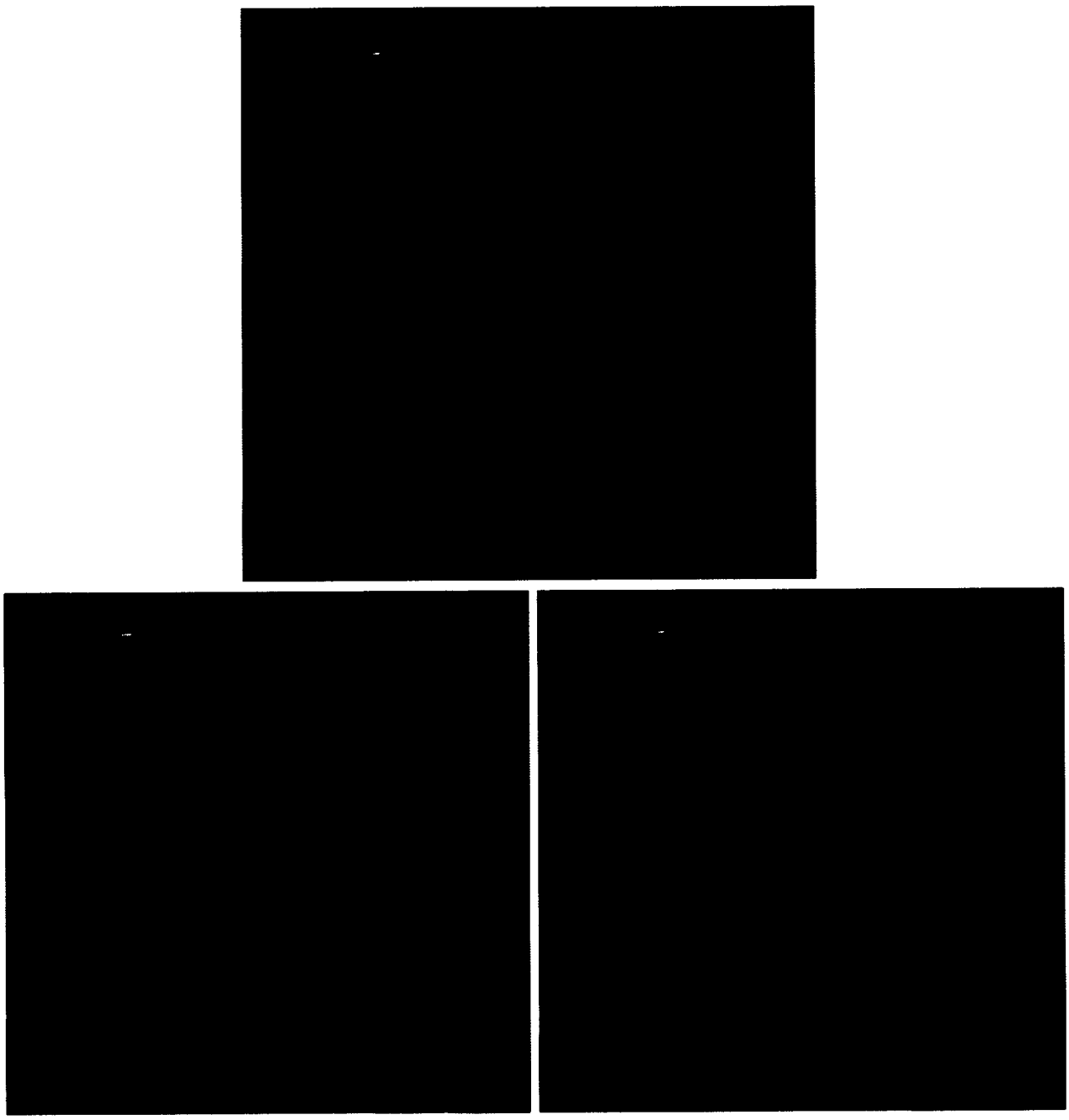

Figure 9: This figure shows a closeup of the *Lockheed fighter* volume. The top volume is not decimated, and the bottom two were generated using the same decimation as in figure 8.

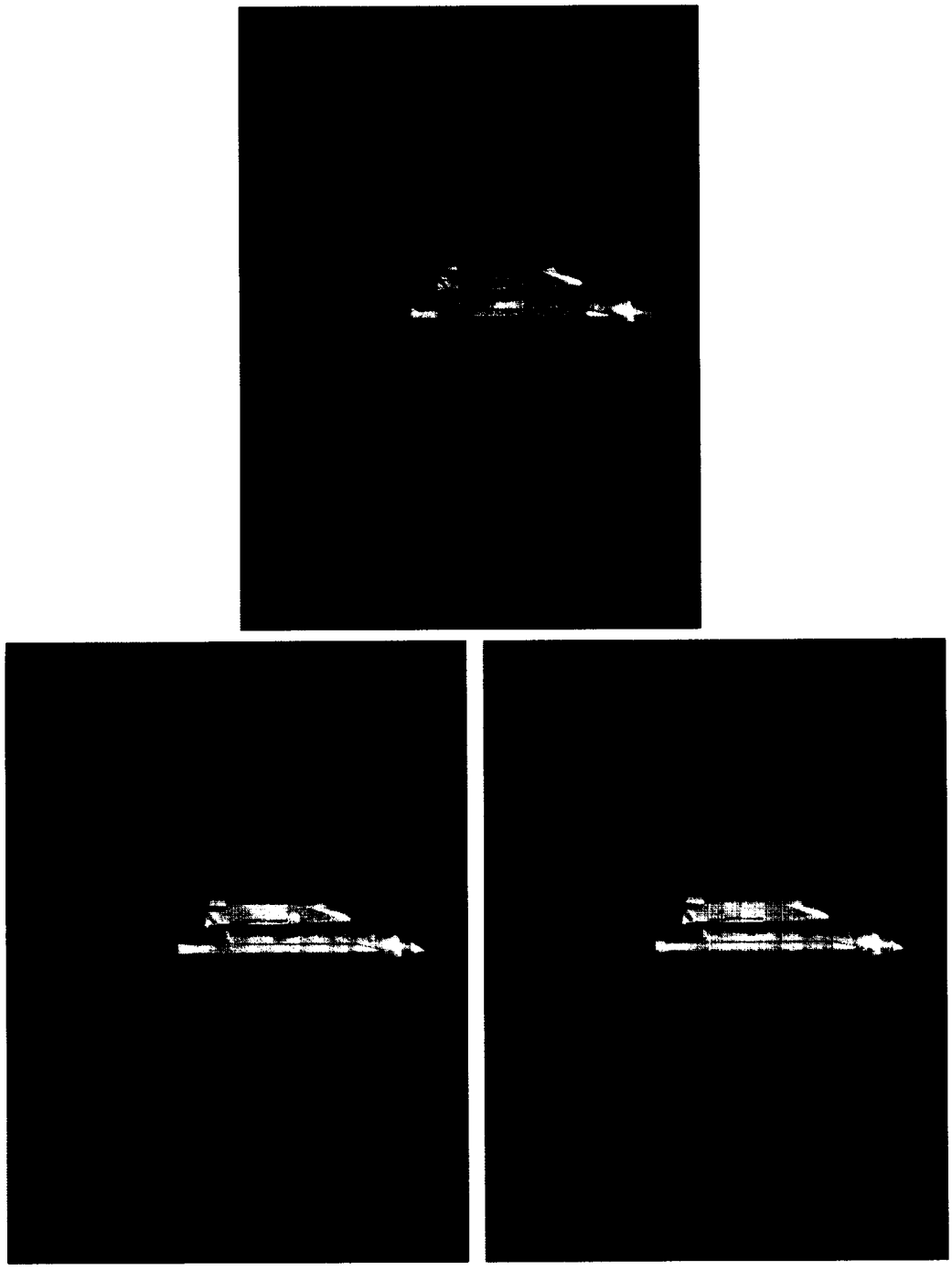

Figure 10: This figure shows the *Space Shuttle* volume. The top image is not decimated. The second image used the transfer function strategy and obtained a speedup of 2.2 times and a decimation of 52%. The third image used the histogram feature strategy and obtained a speedup of 2.6 times and a decimation of 64%.

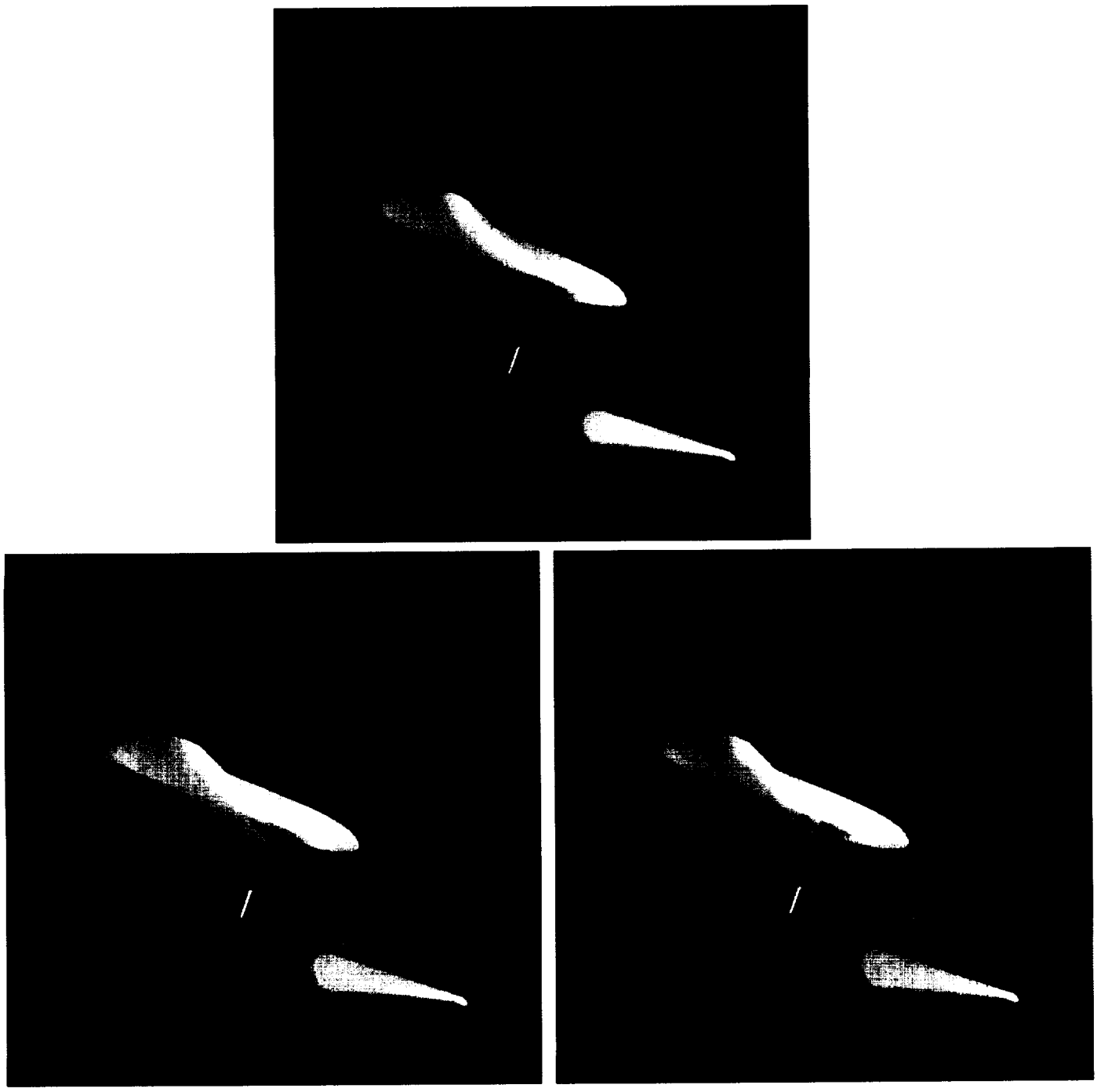

Figure 11: This figure is a closeup of the *Space Shuttle* volume in perspective, showing the nose of the space shuttle along with various other objects, each with their own grid. The top volume is not decimated, and the bottom two were generated using the same decimation as in figure 10.